CLAIMS

What is claimed is:

1. A method comprising:

attempting by a first client to lock a header of a kernel file to prevent simultaneous access by other clients, wherein the kernel file is included in a queue, implemented in a queue data structure, and has one or more slots to accommodate entries in the queue;

obtaining by the first client a lock of the kernel file header;

retrieving by the first client a first unread queue entry from one of the slots;

unlocking by the first client the kernel file header; and

processing by the first client the first unread queue entry.

2. The method of claim 1, wherein:

the queue is a time queue;

each queue entry is a message;

each queue entry file is a message file; and

each queue entry has a queue priority, the queue priority being a delivery time of the queue entry message.

3. The method of claim 1, wherein the queue data structure comprises:

a directory for storing queue entry files; and

a notification file.

4. The method of claim 3, wherein the directory for storing queue entry files is a message directory.

5. The method of claim 1, wherein the kernel file header includes:

a count of the number of empty slots in the kernel file;

a count of the number of slots being filled or processed in the kernel file;

a count of the total number of slots in the kernel file;

an offset of the first slot that is ready to be processed by a client acting as a receiver;

an offset of the first empty slot; and

a queue priority of the first slot that is waiting to be processed.

6. The method of claim 1, wherein the kernel file slots are organized into three virtual groups:

a group for empty slots that are empty and ready to receive a message;

a group for unread slots that each contain an unread message waiting to be processed by a client; and

a group for pending slots that are either being filled by a client or are being processed by a client.

7. The method of claim 1, further comprising:

receiving a new queue entry from a client;

storing the new queue entry in the queue in a list of unread queue entries; and

posting a notification in a notification file.

8. The method of claim 1, wherein the queue data structure is implemented in a file system employing a CIFS protocol.

9. The method of claim 8, wherein the file system is implemented as a native file system on a fault tolerant, network attached storage (NAS) device.

10. The method of claim 9, wherein the NAS device is a RAID device.

11. A method comprising:

locking a header of a kernel file by a first client, wherein the kernel file is included in a time queue and has one or more slots;

inserting into one of the slots an entry which is newer than any other entry in the time queue;

unlocking by the first client the header;

changing an attribute of a notification file, thereby waking a second client; and

locking the header by the second client; and

examining by the second client the entry.

12. The method of claim 11, wherein examining by the second client the entry comprises :

retrieving by the second client the entry;

unlocking the kernel file header; and

processing the entry.

13. The method of claim 11, wherein the kernel file header includes:

a count of the number of empty slots in the kernel file;

a count of the number of slots being filled or processed in the kernel file;

a count of the total number of slots in the kernel file;

an offset of the first slot that is ready to be processed by a client acting as a receiver;

an offset of the first empty slot; and

a queue priority of the first slot that is waiting to be processed.

14. The method of claim 11, wherein the time queue is implemented in a time queue data structure, which is employed in a file system maintaining files on a fault-tolerant platform, the file system coupled for communication with a network operable to perform file operations requested over the network, the file system providing strictly enforced, network-wide file locks.

15. The method of claim 14, wherein the clients are in communication with the network.

16. A computer program product having a computer readable medium having computer program logic recorded thereon comprising:

code for attempting by a first client to lock a header of a kernel file to prevent simultaneous access by other clients, wherein the kernel file is included in a queue, implemented in a queue data structure, and has one or more slots to accommodate entries in the queue;

code for obtaining by the first client a lock of the kernel file header;

code for retrieving by the first client a first unread queue entry from one of the slots;

code for unlocking by the first client the kernel file header; and

code for processing by the first client the first unread queue entry.

17. The computer program product of claim 16, wherein the kernel file header includes:

a count of the number of empty slots in the kernel file;

a count of the number of slots being filled or processed in the kernel file;

a count of the total number of slots in the kernel file;

an offset of the first slot that is ready to be processed by a client acting as a receiver;

an offset of the first empty slot; and

a queue priority of the first slot that is waiting to be processed.